# APPARATUS AND METHOD FOR DATA BYPASS FOR A BI-DIRECTIONAL DATA BUS IN A HUB-BASED MEMORY SUB-SYSTEM

## TECHNICAL FIELD

The present invention relates to memory systems, and more particularly, to memory modules having a data bypass for preventing data collision on a bi-direction data bus.

## BACKGROUND OF THE INVENTION

Computer systems use memory devices, such as dynamic random access memory ("DRAM") devices, to store data that are accessed by a processor. These memory devices are normally used as system memory in a computer system. In a typical computer system, the processor communicates with the system memory through a processor bus and a memory controller. The memory devices of the system memory, typically arranged in memory modules having multiple memory devices, are coupled through a memory bus to the memory controller. The processor issues a memory request, which includes a memory command, such as a read command, and an address designating the location from which data or instructions are to be read. The memory controller uses the command and address to generate appropriate command signals as well as row and column addresses, which are applied to the system memory through the memory bus. In response to the commands and addresses, data are transferred between the system memory and the processor. The memory controller is often part of a system controller, which also includes bus bridge circuitry for coupling the processor bus to an expansion bus, such as a PCI bus.

In memory systems, high data bandwidth is desirable. Generally, bandwidth limitations are not related to the memory controllers since the memory controllers sequence data to and from the system memory as fast as the memory devices allow. One approach that has been taken to increase bandwidth is to increase the speed of the memory data bus coupling the memory controller to the memory devices. Thus, the same amount of

information can be moved over the memory data bus in less time. However, despite increasing memory data bus speeds, a corresponding increase in bandwidth does not result. One reason for the non-linear relationship between data bus speed and bandwidth is the hardware limitations within the memory devices themselves. That is, the memory controller has to schedule all memory commands to the memory devices such that the hardware limitations are honored. Although these hardware limitations can be reduced to some degree through the design of the memory device, a compromise must be made because reducing the hardware limitations typically adds cost, power, and/or size to the memory devices, all of which are undesirable alternatives. Thus, given these constraints, although it is easy for memory devices to move "well-behaved" traffic at ever increasing rates, for example, sequel traffic to the same page of a memory device, it is much more difficult for the memory devices to resolve "badly-behaved traffic," such as bouncing between different pages or banks of the memory device. As a result, the increase in memory data bus bandwidth does not yield a corresponding increase in information bandwidth.

In addition to the limited bandwidth between processors and memory devices, the performance of computer systems is also limited by latency problems that increase the time required to read data from system memory devices. More specifically, when a memory device read command is coupled to a system memory device, such as a synchronous DRAM ("SDRAM") device, the read data are output from the SDRAM device only after a delay of several clock periods. Therefore, although SDRAM devices can synchronously output burst data at a high data rate, the delay in initially providing the data can significantly slow the operating speed of a computer system using such SDRAM devices. Increasing the memory data bus speed can be used to help alleviate the latency issue. However, as with bandwidth, the increase in memory data bus speeds do not yield a linear reduction of latency, for essentially the same reasons previously discussed.

Although increasing memory data bus speed has, to some degree, been successful in increasing bandwidth and reducing latency, other issues are raised by this

approach. For example, as the speed of the memory data bus increases, loading on the memory bus needs to be decreased in order to maintain signal integrity since traditionally, there has only been wire between the memory controller and the memory slots into which the memory modules are plugged. Several approaches have been taken to accommodate the increase in memory data bus speed. For example, reducing the number of memory slots, adding buffer circuits on a memory module in order to provide sufficient fanout of control signals to the memory devices on the memory module, and providing multiple memory device interfaces on the memory module since there are too few memory module connectors on a single memory device interface. The effectiveness of these conventional approaches are, however, limited. A reason why these techniques were used in the past is that it was cost-effective to do so. However, when only one memory module can be plugged in per interface, it becomes too costly to add a separate memory interface for each required memory slot. In other words, it pushes the system controllers package out of the commodity range and into the boutique range, thereby, greatly adding cost.

One recent approach that allows for increased memory data bus speed in a cost effective manner is the use of multiple memory devices coupled to the processor through a memory hub. In a memory hub architecture, or a hub-based memory sub-system, a system controller or memory controller is coupled over a high speed bi-directional or unidirectional memory controller/hub interface to several memory modules. Typically, the memory modules are coupled in a point-to-point or daisy chain architecture such that the memory modules are connected one to another in series. Thus, the memory controller is coupled to a first memory module, with the first memory module connected to a second memory module, and the second memory module coupled to a third memory module, and so on in a daisy chain fashion.

Each memory module includes a memory hub that is coupled to the memory controller/hub interface and a number of memory devices on the module, with the memory hubs efficiently routing memory requests and responses between the controller and the memory devices over the memory controller/hub interface. Computer systems employing

this architecture can use a high-speed memory data bus since signal integrity can be maintained on the memory data bus. Moreover, this architecture also provides for easy expansion of the system memory without concern for degradation in signal quality as more memory modules are added, such as occurs in conventional memory bus architectures.

Although computer systems using memory hubs may provide superior performance, they may often fail to operate at optimum efficiency for a variety of reasons. One such reason is the issue of managing data collision between data flowing to and from the memory controller through the memory hubs. In conventional memory controllers, one approach taken to avoid data collision is to delay the execution of one memory command until the completion of another memory command. For example, with a conventional memory controller, a write command issued after a read command is not allowed to begin until the read command is nearly completed in order to avoid the read (i.e., inbound) data colliding with the write (i.e., outbound) data on the memory bus. However, forcing the write command to wait effectively reduces bandwidth, which is inconsistent with what is typically desired in a memory system.

## SUMMARY OF THE INVENTION

One aspect of the present invention is directed to a memory hub having a data bypass circuit. The memory hub includes first and second link interfaces for coupling to respective data busses, a data path coupled to the first and second link interfaces and through which data is transferred between the first and second link interfaces. The memory hub further includes a write bypass circuit coupled to the data path for coupling write data on the data path and temporarily storing the write data to allow read data to be transferred through the data path while the write data is temporarily stored. In another aspect of the invention, a method for writing data to a memory location in a memory system coupled to a memory bus is provided. The method includes accessing read data in the memory system, providing write data to the memory system on the memory bus, and coupling the write data to a register for temporary storage of the write data. While the data is temporarily stored,

the read data is coupled from the memory bus and provided for reading. The write data is recoupled to the memory bus and written to the memory location.

## BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a block diagram of a computer system having memory modules in a memory hub architecture in which embodiments of the present invention can be implemented.

Figure 2 is a partial block diagram of a memory hub according to an embodiment of the present invention for use with the memory modules of Figure 1.

Figure 3 is a block diagram of a data bypass circuit for the memory hub of Figure 2 according to an embodiment of the present invention.

Figure 4 is a block diagram illustrating the operation of the data bypass circuit of Figure 3 for a computer system having the memory hub architecture of Figure 1 and the memory hub of Figure 2.

## DETAILED DESCRIPTION OF THE INVENTION

Embodiments of the present invention are directed to a memory hub having bypass circuitry that provides data bypass for a bi-directional data bus in a hub-based memory sub-system. Certain details are set forth below to provide a sufficient understanding of various embodiments of the invention. However, it will be clear to one skilled in the art that the invention may be practiced without these particular details. In other instances, well-known circuits, control signals, and timing protocols have not been shown in detail in order to avoid unnecessarily obscuring the invention.

Figure 1 illustrates a computer system 100 according to one embodiment of the present invention. The computer system 100 includes a processor 104 for performing various computing functions, such as executing specific software to perform specific calculations or tasks. The processor 104 includes a processor bus 106 that normally includes an address bus, a control bus, and a data bus. The processor bus 106 is typically

coupled to cache memory 108. Typically, the cache memory 108 is provided by a static random access memory ("SRAM"). The processor bus 106 is also coupled to a system controller 110, which is sometimes referred to as a bus bridge.

The system controller 110 serves as a communications path to the processor 104 for a variety of other components. For example, as shown in Figure 1, the system controller 110 includes a graphics port that is typically coupled to a graphics controller 112. The graphics controller is typically coupled to a video terminal 114, such as a video display. The system controller 110 is also coupled to one or more input devices 118, such as a keyboard or a mouse, to allow an operator to interface with the computer system 100. Typically, the computer system 100 also includes one or more output devices 120, such as a printer, coupled to the processor 104 through the system controller 110. One or more data storage devices 124 are also typically coupled to the processor 104 through the system controller 110 to allow the processor 104 to store data or retrieve data from internal or external storage media (not shown). Examples of typical storage devices 124 include hard and floppy disks, tape cassettes, and compact disk read-only memories (CD-ROMs).

The system controller 110 includes a memory hub controller 128 that is coupled to memory hubs 140 of several memory modules 130a, 130b, 130c, . . . 130n. The memory modules 130 serve as system memory for the computer system 100, and are preferably coupled to the memory hub controller 128 through a high-speed bi-directional memory controller/hub interface 134. The memory modules 130 are shown coupled to the memory hub controller 128 in a point-to-point arrangement in which the memory controller/hub interface 134 is coupled through the memory hubs 140 of the memory modules 130. That is, the memory controller/hub interface 134 is a bi-directional bus that couples the memory hubs 140 in series. Thus, information on the memory controller/hub interface 134 must travel through the memory hubs 140 of "upstream" memory modules 130 to reach a "downstream" destination. For example, with specific reference to Figure 1, information transmitted from the memory hub controller 128 to the memory hub 140 of the

memory module 130c will pass through the memory hubs 140 of the memory modules 130a and 130b.

It will be appreciated, however, that topologies other than the point-to-point arrangement of Figure 1 may also be used. For example, a coupling arrangement may be used in which a separate high-speed link (not shown) is used to couple each of the memory modules 130 to the memory hub controller 128. A switching topology may also be used in which the memory hub controller 128 is selectively coupled to each of the memory modules 130 through a switch (not shown). Other topologies that may be used will be apparent to one skilled in the art. Additionally, the memory controller/hub interface 134 coupling the memory modules to the memory hub controller may be an electrical or optical communication path. However, other types of communications paths can be used for the memory controller/hub interface 134 as well. In the event the memory controller/hub interface 134 is implemented as an optical communication path, the optical communication path may be in the form of one or more optical fibers. In such case, the memory hub controller 128 and the memory modules will include an optical input/output port or separate input and output ports coupled to the optical communication path, as well known in the art.

The memory hubs 140 control access to memory devices 148 of the respective memory module 130. In Figure 1, the memory devices are illustrated as synchronous dynamic random access memory ("SDRAM") devices. However, memory devices other than SDRAM devices may also be used. As also shown in Figure 1, the memory hub is coupled to four sets of memory devices 148 through a respective memory bus 150. Each of the sets includes four memory devices 148 for a total of 20 memory devices 148 for each memory module 130. The memory busses 150 normally include a control bus, an address bus, and a data bus, as known in the art. However, it will be appreciated by those ordinarily skilled in the art that other bus systems, such as a bus system using a shared command/address bus, may also be used without departing from the scope of the present invention. It will be further appreciated that the arrangement of the

memory devices 148, and the number of memory devices 148 can be modified without departing from the scope of the present invention.

Figure 2 illustrates a portion of the memory hub 140 according to an embodiment of the present invention. The memory hub 140 includes a local hub circuit 214 coupled to the memory controller/hub interface 134 (Figure 1). The local hub circuit 214 is further coupled to memory devices 148 through the memory bus 150. The local hub circuit 214 includes control logic for processing memory commands issued from the memory controller 128 and for accessing the memory devices 148 over the memory bus 150 to provide the corresponding data when the memory command is directed to the respective memory module 130. The design and operation of such control logic is well known by those ordinarily skilled in the art, and consequently, a more detailed description has been omitted from herein in the interest of brevity. The memory hub 140 further includes a data bypass circuit 286 coupled to the local hub circuit 214. As will be explained in more detail below, the data bypass circuit 286 is used to temporarily capture data passing to a distant memory hub, which allows data returning from another distant memory hub to pass through the memory hub 140 before the captured data continues onto the distant memory hub. Thus, the data bypass circuit 286 provides a data bypass mechanism that can be used to avoid data collisions on the bi-directional memory controller/hub interface 134 to which the memory hub 140 is coupled.

As previously discussed, one approach taken by conventional memory sub-systems to avoid data collision is to delay the execution of one memory command until the completion of another memory command. For example, in typical memory systems a write command issued after a read command would not have been allowed to start until near the completion of the read command in order to avoid the read (i.e., inbound) data colliding with the write (i.e., outbound) data on the memory controller/hub interface 134. In contrast, by employing the memory hub 140 having the data bypass circuit 286, write commands issued after a read command can be sequenced earlier than compared with

conventional memory systems, and consequently, memory commands scheduled after the earlier scheduled write command can be executed sooner as well. .

Figure 3 illustrates a data bypass circuit 300 according to an embodiment of the present invention. The data bypass circuit 300 can be substituted for the data bypass circuit 286 (Figure 2) and can be implemented using conventional designs and circuits well known to those ordinarily skilled in the art. The data bypass circuit 300 includes an input buffer 302 that receives input write data WR-DATA_IN and provides the same to a bypass register/FIFO 304 and a first input of a multiplexer 306. An output of the bypass register/FIFO 304 is coupled to a second input of the multiplexer 306. Selection of which of the two inputs to couple to the output of the multiplexer 306 is made by an enable signal EN generated by a bypass select logic 308. The EN signal is also provided to an input/output buffer 310 as an output enable signal activating or deactivating the input/output buffer 310. The bypass select logic 308 generates the appropriate EN signal in response to an activation signal BYPASS_EN provided by the memory hub controller 128 (Figure 1). Alternatively, the BYPASS_EN signal may be provided from other memory hubs (not shown) that are part of the same memory system. The circuitry of the data bypass circuit is conventional, and it will be appreciated that the circuits of the data bypass circuit 300 can be implemented using conventional designs and circuitry well known in the art.

In operation, WR_DATA_IN received by the data bypass circuit 300 is driven through the input buffer 302 and is provided to the first input of the multiplexer 306. The WR_DATA_IN is also saved in the bypass register/FIFO 304. In response to an inactive BYPASS_EN signal, an active EN signal is generated by the bypass select logic 308. The active EN signal enables output by the input/output buffer 310 and couples the output of the input buffer 302 to the input of the input/output buffer 310 through the multiplexer 306. As a result, the WR_DATA_IN is provided directly to the input of the input/output buffer 310 and the WR_DATA_IN is provided through the data bypass circuit 300 without any bypass. However, in response to an active BYPASS_EN signal, the bypass select logic 308 generates an inactive EN signal, disabling the output function of the

input/output buffer 310 and placing its output in a high-impedance state. Additionally, the inactive EN signal couples the input of the input/output buffer 310 to the output of the bypass register/FIFO 304. In this manner, the WR_DATA_IN is received by the data bypass circuit 300, stored by the bypass register/FIFO 306, and applied to the input of the input/output buffer 310. However, due to the inactive state of the EN signal, the WR_DATA_IN is not provided as output data WR_DATA_OUT by the input/output buffer 310. As a result, the WR_DATA_IN is held in a bypass state until the BYPASS_EN signal becomes inactive, at which time, the EN signal become active again, enabling the input/output buffer 310 to provide the WR_DATA_IN as WR_DATA_OUT data. The multiplexer 306 is also switched back to coupling the output of the input buffer 302 directly to the input of the input/output buffer 310 to allow WR_DATA_IN to pass through the data bypass circuit unhindered.

Operation of the data bypass circuit 286 will be described with reference to Figure 4. Figure 4 is similar to Figure 1, except that Figure 4 has been simplified. In particular, many of the functional blocks of Figure 1 have been omitted, with only the memory modules 130a-130c being shown, and represented by memory hubs 140a-140c. Only one memory device 148a-148c is shown to be coupled to a respective memory hub 140a-140c through a respective memory bus 150a-150c. As with Figure 1, the memory hubs 140a-140c are coupled by a high-speed bi-directional memory controller/hub interface 134 to a memory hub controller 128.

In Figure 4, it is assumed that the memory hub controller 128 has just issued read and write commands, with the read command sequenced prior to the write command. The read command is directed to the memory module 130b and the write command is directed to the memory module 130c. That is, the memory module to which data will be written is further downstream than the memory module from which data is read. In response to the read command, the memory hub 140b begins retrieving the read data (RD) from the memory device 148b, as indicated in Figure 4 by the "(1)". With the read command issued, the write command is then initiated, and the write data (WD) is provided

onto the memory controller/hub interface 134. However, since the memory hub controller 128 is expecting the RD to be returned from the memory module 130b, the memory hub 140a is directed to capture the WD in its data bypass circuit 286a. As a result, the memory hub 286a captures the WD to clear the memory controller/hub interface 134, as indicated in Figure 4 by the "(2)", for the RD to be returned to the memory hub controller 128. When the memory hub 140b has retrieved the RD from the memory device 148b, the RD is then provided to the memory hub controller 128 through the memory controller/hub interface 134, as indicated in Figure 4 by the "(3)" to complete the read request. Upon the RD passing through the memory hub 140a on its way to the memory hub controller 128, the memory hub 140a releases the WD from the data bypass circuit 286a to continue its way to the memory hub 140c. The WD is provided to the memory hub 140c through the high-speed link, which is now clear between the memory hub 140a and 140c. Upon reaching the memory hub 140c, the WD is written in the memory device 148c, as shown in Figure 4 by the "(4)". In an embodiment of the present invention, coordination of the data flow of the RD and WD on the memory controller/hub interface 134 and through the data bypass circuits 286 is under the control of the memory hub controller 128. For example, in the previous example the memory hub controller ensures that any WD flowing in the opposite direction of the RD is out of the way when retrieving RD from the memory module 130b. It will be appreciated, however, that in alternative embodiments data flow through the memory controller/hub interface 134 and the data bypass circuits 286 can be managed differently, such as the memory hub controller 128 sharing coordination of the data flow with the memory hubs 140.

In the previous example, the RD is returned to the memory hub controller 128 as in a conventional memory system. That is, the RD transmitted by the memory devices 148 is provided to the memory controller without any significant delay. However, by employing the previously described data bypass mechanism, write commands can be scheduled earlier than with conventional memory systems. A write command issued after a read command would not have been allowed to start until near the completion of the read

command in typical memory systems. In contrast, embodiments of the present invention allow a subsequently issued write command to be scheduled earlier, thus, reducing the time gap between read and write commands. As a result, commands scheduled behind an earlier scheduled write command have an overall reduced latency.

From the foregoing it will be appreciated that, although specific embodiments of the invention have been described herein for purposes of illustration, various modifications may be made without deviating from the spirit and scope of the invention. Accordingly, the invention is not limited except as by the appended claims.